NEW METHODS FOR PSEUDOEXHAUSTIVE TESTING

E. SELÉNYI

Department of Measurement and Instrumentation Engineering,
Technical University, H-1521 Budapest

Received by June 5, 1986
Presented by Prof. Dr. L. Schnell

Abstract

Pseudoexhaustive testing of combinational circuits has become of great importance recently. These methods are keeping most of the benefits of the classical exhaustive testing which check every combination of the input signals, but they need a considerably shorter sequence of test patterns. In this paper we give a survey of pseudoexhaustive testing. Two new code construction methods are presented: a systematic procedure to generate an effective exhaustive code for every two dimensional subspace of the inputs; and an extension of the codes from the k dimensional space to $k+1$. The efficiency of the new methods is compared to the ones described in the literature.

Introduction

Logic testing of digital circuits has always been of central interest because of the high reliability required. Testing of LSI and VLSI circuits however, is getting increasingly difficult as both circuits density and the manufactured quantity increase.

Traditional test techniques require the derivation of the input test sets with the associated output responses based on a fault model of the device under test, so, that the faults contained by the model could be detected (or perhaps diagnosed). These test sets are then programmed into an automatic tester, which applies them to the unit under test and checks the responses. There are a number of difficulties with this approach [1].

1) A fault model is required. In LSI and VLSI circuits the classical assumption that only single stuck-at-faults have to be considered may no longer be valid. More complex models substantially increases the difficulty of pattern generation.

2) Pattern generation is required. Automatic test pattern generation is expensive and does not provide sufficiently high fault coverage. Manual methods are restricted in complexity and they considerably increase the production cycle.

3) An expensive tester is needed.

Much effort has been made all over the world to avoid these problems. To refine the classical method new and more effective test generating algorithms have been developed, more productive testers have been designed, etc.

Besides these advances two kinds of new approaches have been worked out recently. One of them is to design circuits to facilitate testing. This means structural restrictions for the designer, but testing of such circuits is simple. A typical example of this method is LSSD.

The other approach named pseudoexhaustive testing has been worked out by E. J. McCluskey et al. This paper contains a survey of the results in pseudoexhaustive testing, and besides, some new code construction methods will be introduced worked out by the author.

Pseudoexhaustive testing

The traditional exhaustive test of a combinational circuit includes every combination of the input set. thus except for those bridging-faults that lead to an asynchronous sequential circuit, the unit is completely tested without a fault model.

The disadvantage of this method is that for n inputs 2^n test vectors are required. In practice only circuits realized by ROM need this kind of testing; the gate-structured circuits can be exhaustively tested by a considerably shorter sequence of test patterns. This simplification can be achieved by partitioning the circuit into subcircuits which are then separately tested exhaustively.

Figure 1 shows a circuit, where $F1$, $F2$ and $F3$ are arbitrarily realized logical functions. The circuit has six inputs, so its exhaustive test would contain 64 test vectors. In this case, however the number of the test vectors can be reduced, as each output depends only on 4 of the inputs. Thus, except for the bridging of the outputs, both the $F1$–$F2$ and the $F1$–$F3$ subcircuits can be tested with 16–16 vectors. By an appropriate choice of input vectors (e.g. $a = \bar{e}$,

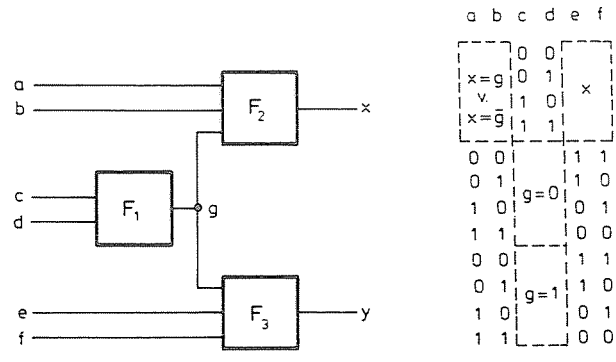

Fig. 1

$b = \bar{f}$) these two sequences can be applied concurrently, thus the whole circuit can be tested within 16 steps.

The test sequence can be further reduced by the serial decomposition of $F1$–$F2$ ($F1$–$F3$). Four test vectors and the sensitization of g through $F2$ are sufficient for testing $F1$, $F2$ and $F3$, both having 3 inputs to be tested with 8 vectors. If the logic functions of $F1$, $F2$, $F3$ are known, surely some of the first four vectors can be applied concurrently with some of the second eight ones, thus the test would contain less then 12 steps.

With this decomposition method we can
— check every single stuck-at-fault in the circuit,
— check every multiple stuck-at-fault and bridging-fault within the modules, but
— we might not detect the multiple stuck-at- and bridging-faults between the modules.

Methods for an adequate decomposition are described in the literature [1], [3], [8], [11]. As we have shown in the example, the decomposition of the gate-structured circuit led to a testing, which was not exhaustive on the total n dimensional space, but only on its k_1, k_2,... k_m — not necessarily disjoint subspaces.

It is possible to find the proper k_i-subspaces by a thorough examination of any concrete circuit, but usually it is much easier to determine the maximum size ($k_{max} = k$) of them, hence a test being exhaustive for all k-subspaces of the n dimensional space of inputs can be used.

The problem in general is solved by the $\langle n, k, T \rangle$ pseudoexhaustive codes. These are code sets containing T n-dimensional vectors which exhaustively cover every k-subspace.

The pseudoexhaustive codes can be efficiently used also for testing registers and memories, where the faults are usually caused by cells stuck together according to some logic function. RAM tests, which check every pair of cells in all four combinations are widely used.

Pseudoexhaustive codes

Up to now no general solution exists for finding an optimal $\langle n, k, T \rangle$ code. An optimal solution means practically the minimum number (T) of necessary test vectors for given n and k. Pseudoexhaustive codes described in the literature are not aimed to be optimal, only to be more or less effective. In the following, we will describe some of the interesting code constructions.

Optimal code for $k=n-1$ [4]

Code construction: every combination of the $k=n-1$ bits are realized in $T=2^k$ rows. The n^{th} column is calculated as the mod2 sum of the other columns (see Fig. 2).

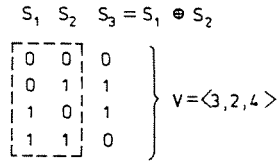

Fig. 2

Optimal code for $k=2$ [16]

In case of $k=2$ the all-zero code (v_0) and the all-one code (v_1) contain the combinations 00 and 11 for each pair of bits. To get the combinations 01 and 10, any two columns must be unordered (a and b binary vectors are unordered, if there exists such an i^{th} bit position, that $a_i=0$ and $b_i=1$, and a j^{th} one, that $a_j=1$ and $b_j=0$).

An effective way of creating unordered codes is to choose codes of the same weight.

Thus the optimal code construction (see Fig. 3) is the following: Let the first element of any s_i column code be 0. The next $T-1$ elements ($T-1$ being an odd number) are forming a code of weight $T/2$.

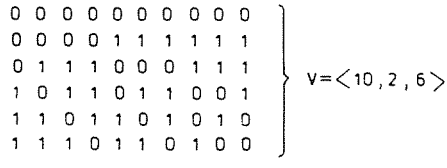

Fig. 3

The maximum number of such columns $n_{\max}=\binom{T-1}{T/2}$, this is the maximum length of the codes. This method provides the 11 combinations without containing v_1.

Systematic code for $k=2$

Code construction: Let the i^{th} column code be

$$s_i=0\langle i\rangle 1\langle i\rangle$$

where $\langle i \rangle$ is the binary expression for i, $\langle \bar{i} \rangle$ is the 1's complement of $\langle i \rangle$ (See Fig. 4.). Having T(Tbeing an even number) test vectors, $n_{\max} = 2^{T/2-1}$ bits can be coded.

S_i	S_0	S_1	S_2	S_3
0	0	0	0	0
↑ i	0	0	1	1
↓	0	1	0	1
1	1	1	1	1
↑ ī	1	1	0	0
↓	1	0	1	1

Fig. 4

Constant weight codes [2]

Code construction: Let $V = \langle n, k, T \rangle$ be a code containing every vector v_i of weight(s) w such, that $w = c \mod (n-k+1)$ for a constant c, where $0 \leq c \leq n-k$. For $k = n-1$ this method provides the optimal code. I.e. in this case $w = c \bmod 2$, if $c = 0$, then all codes of even weights, if $c = 1$, then all codes of odd weights are included in V.

An example for $k = 2$:

Let	$c = 0$,	then	$w_1 = 0$,	$w_2 = n-1$
Let	$c = 1$,	then	$w_1 = 1$,	$w_2 = n$
Let	$c = 2$,	then	$w = 2$	
Let	$c = n-2$,	then	$w = n-2$.	

The best cases of these are when $c = 0$ or $c = 1$. If $c = 0$, then Vincludes the all-zero vector, and all the vectors containing only one zero. In this case $T = n + 1$, which is considerably worse than the optimal solution. For $k = 3$ the best choice is to take all the codes containing a single zero, and all the codes containing single one ($w_1 = 1$, $w_2 = n-1$).

The number of the necessary vectors $T = 2k$. (See table 1.)

We finish the survey of the code construction methods described in the literature by mentioning [6], where cyclic codes are used in an interesting way to provide pseudoexhaustive codes. This method is not more effective than others mentioned above for $k \leq 3$, but for $k > 3$ some efficient solutions were provided. The algorithm is rather complex, and (at least up to now) the value of n cannot be arbitrary.

Effective systematic code for $k = 2$

Let us consider a $V_0 = \langle n_0, 2, T_0 \rangle$ pseudoexhaustive code! A code for arbitrary n and $k = 2$ can be obtained from V_0 by the following algorithm: Let us remove from V_0 the all-zero code and the all-one code (if they are included):

$V'_0 = V_0 - \{v_0, v_1\}$. Let us denote the columns of V'_0 with $s_0, \ldots s_{n0-1}$, respectively. To construct the i^{th} column code in the n dimensional space, let us rewrite i in the number system based on n_0:

$$i = \sum_{p=0}^{q} a_p \cdot n_0^p; \qquad 0 \leq a_p \leq n-1.$$

Let $s'_i = \langle sa_q \rangle \langle sa_{q-1} \rangle \ldots \langle sa_0 \rangle$; $0 \leq i \leq n-1$.

Let us complete the code determined by the s'_i columns with the v_0, v_1 vectors (if they were removed before).

Theorem 1. The resulting code is a $V = \langle n, 2, T \rangle$ pseudoexhaustive code.

Proof: If $i \neq j$, then their forms in the number system based on n_0 will differ from each other in at least one position. Let us denote this position with p. Thus the p segment will contain every combination in s'_i and s'_j, provided by V'_0. Considering also the probably previously removed v_0 and v_1 codes, the resulting V provides every combination included in V_0 for any two bits. As V_0 exhaustively covers every $k=2$ subspace, so does V, too. Q.E.D.

	S_{00}	S_{01}	S_{02}	S_{10}	S_{11}	S_{12}	S_{20}	S_{21}	S_{22}	S_{ij}
	0	0	0	0	0	0	0	0	0	0
	0	1	1	0	1	1	0	1	1	↑
V_0	1	0	1	1	0	1	1	0	1	S_{aj}
	1	1	0	1	1	0	1	1	0	↓
	0	0	0	1	1	1	1	1	1	↑
	1	1	1	0	0	0	1	1	1	S_{ai}
	1	1	1	1	1	1	0	0	0	↓

Fig. 5

This method is tendentiously optimal, if the optimal code set $V_0 = \langle 3, 2, 4 \rangle$ is chosen initially. In this case using $T = 3t+1$ (t being an integer) vectors the maximum length of the code vectors $n_{\max} = 3^{(T-1)/3}$. This method is getting more and more effective as T increases, compared to the systematic code for $k=2$ described in the literature. An example is shown in Fig. 5.

Extending pseudoexhaustive codes from k to $(k+1)$

Lemma 1. Any projection (V_k^{k+1}) of a pseudoexhaustive code $V_k = \langle n, k, T \rangle$, $n > k$, onto a $k+1$ dimensional subspace contains at least one vector of any pair of one Hamming distance.

Proof: Suppose, that V_k^{k+1} contains none of the following two vectors of one Hamming distance:

$$V_{\alpha 1} = a_1, a_2, \ldots a_{i-1}, a_i, a_{i+1}, \ldots a_{k+1} \quad \text{and}$$

$$V_{\alpha 2} = a_1, a_2, \ldots a_{i-1}, \overline{a_i}, a_{i+1}, \ldots a_{k+1}.$$

Removing the i^{th} bit, the resulting projection V_k^k does not contain the combination

$$V_\alpha = a_1, a_2, \ldots a_{i-1}, a_{i+1}, \ldots a_{k+1},$$

thus V_k is not covering exhaustively every k-subspace, which is a contradiction. Q.E.D.

Lemma 2. If V_k^{k+1} includes codes of both even and odd weights, then it contains also codes of one Hamming distance.

Proof: According to Lemma 1 at least one of any two codes of one Hamming distance are included in V_k^{k+1}. If there are no two codes of one Hamming distance in V_k^{k+1}, then it contains only either codes of odd weights or codes of even weights, which is a contradiction. Q.E.D.

Theorem 2. Let $V_k = \langle n, k, T \rangle$ be a pseudoexhaustive code including v_0 and $O_{k+1} = \langle n, k+1, t \rangle$ such a code set, that every projection of O_{k+1} on a $k+1$ space contains at least one code of odd weight, then

$$V_{k+1} = (V_k \oplus V_k) U (V_k \oplus O_{k+1}),$$

where $V_{k+1} = \langle n, k+1, T_{k+1} \rangle$ is a pseudoexhaustive code set. (The mod2 operation between code sets has to be executed between pairs of codes.)

Proof: The projection of V_k onto the $k+1$-space will always contain a code of even weight (the projection of v_0). Therefore the following two cases must be considered:

a) All elements of V_k^{k+1} are of even weights. Then according to Lemma 1 every code of even weight is contained in V_k^{k+1} and because of $V_k \oplus V_k$, also in V_{k+1}^{k+1}. As O_{k+1}^{k+1} projected on the same space contains at least one code of odd weight, $V_k^{k+1} \oplus O_{k+1}^{k+1}$ contains every code of odd weight.

b) V_k^{k+1} includes codes of both even and odd weights. In this case according to Lemma 2 there are two codes of one Hamming distance in it:

$$V_{\alpha 1} = a_1, a_2 \ldots a_i \ldots a_{k+1}$$

$$V_{\alpha 2} = a_1, a_2 \ldots \overline{a_i} \ldots a_{k+1}$$

Suppose, that V_{k+1}^{k+1} doesn't contain the code v_β, i.e. it isn't exhaustive for $k+1$ then as $V_k^{k+1} \oplus V_k^{k+1}$ is included in V_{k+1}^{k+1}, V_k^{k+1} contains neither the code $v_{\gamma 1} = v_{\alpha 1} \oplus v_\beta$, nor $v_{\gamma 2} = v_{\alpha 2} \oplus v_\beta$, where $v_{\gamma 1}$ and $v_{\gamma 2}$ are codes of one Hamming distance. This contradicts Lemma 1. Q.E.D.

The efficiency of the recoursive extension for k depends basically on the construction of O_{k+1}. If k is an even number, then O_{k+1} consists only of one code: v_1. For the case of odd k the optimal solution (the minimum number of elements) has not been found yet.

Theorem 3. If k is an even number and from the fact that v_i is an element of V_k comes that so is $\bar{v}_i$ (V_k contains only pairs of inverted codes) then the code set $V_{k+1}=V_k \oplus V_k$ is exhaustive for $k+1$.

Proof: Any projection of V_k onto the $k+1$ dimensional space contains codes of both odd and even weights (because of the inverted pairs), therefore the proof of Theorem 2b. can be applied. Q.E.D.

Effective code constructions for $k=3$

Several V_3 code sets can be constructed by applying the extension method on different V_2 code sets. In the following we are going to show two procedures.

K1 construction

Let $V_2=\langle n, 2, T_2\rangle$ be the systematic code containing columns of the form $s_i=0\langle i\rangle 1\langle i\rangle$. This code set consists of inverted pairs of codes, thus the construction according to Theorem 3 can be used. Applying the following equalities:

$$v_i \oplus v_i = v_0 \qquad \text{and} \qquad v_i \oplus \bar{v}_i = v_1, \quad v_i \oplus v_j = \bar{v}_i \oplus \bar{v}_j,$$

the number of elements in $V_3=\langle n, 3, T_3\rangle$ will be much less then $T_2^2/2$:

$$T_3 \leq 2+2\binom{T_2/2}{2}$$

Making use of the equation between T_2 and n, the number of the needed test vectors for given n

$$T_3 \leq 2+\lceil \mathrm{ld} n \rceil + \lceil \mathrm{ld} n \rceil^2$$

The values of T_3 for small values of n are given in table 1.

K2 construction

Let $V_2=\langle n, 2, T_2\rangle$ be an optimal code set, containing v_0, and O containing the code v_1. Let us apply the construction of Theorem 2. Thus

$$n_{\max}=\binom{T_2-2}{\left[\frac{T_2-2}{2}\right]} \quad \text{and} \quad T_3=2T_2-2+\binom{T_2-3}{2}$$

The derivation of this formula is omitted here. There is no explicit equation between n and T_3, but for small values of n, table 1 contains the counted results.

Comparison of the codes for $k=3$

In table 1 we compared the effectiveness of the two new methods to the

Table 1

n	T_3		
	$K1$	$K2$	$K3$
3			8
4	8		8
5			10
6		13	12
7			14
8	14		16
9			18
10		18	20
11			22
12			24
16	22		32
20		24	40
27			54
32	32		64
35		31	70
64	44		128
70		39	140

constant weight method ($K3$). It can be seen that the systematic-like $K1$ construction is better for large values of n, $n>11$, the non-systematic $K2$ construction is better than $K3$ for $n>12$, and the difference grows rapidly by increasing the value of n.

References

1. McCluskey, E. J.–Bozorgui-Nesbat, S.: Design for Autonomous Test. IEEE. Trans. Comput. vol. C-*30*, No. 11. 866–875 (1981).
2. Tang, D. T.–Woo, L. S.: Exhaustive Test Pattern Generation with Constant Weight Vectors. IEEE Trans. Comput. vol. C-*32*., No. 12. 1145–1150 (1983).
3. McCluskey, E. J.: Verification Testing — A Pseudoexhaustive Test Technique. IEEE Trans. Comput. vol. C-*33*, No. 6 (1984).
4. McCluskey, E. J.: Verification Testing. 19th Annual Design Automation Conference. 495–500, Las Vegas, Nevada, June 14–16, 1982.

5. McCLUSKEY, E. J.: Exhaustive and Pseudo-Exhaustive Test. International Test Conference' 83. Tutorial on Built-in Test. Philadelphia, Pennsylvania, Oct. 17. 1983.
6. TANG, D. T.–CHEN, C. L.: Logic Test Pattern Generation Using Linear Codes. IEEE Trans. Comput. vol. C-*33*, No. 9., 845–849 (1984).
7. TANG, D. T.–CHEN, C. L.: Logic Test Pattern Generation Using Linear Codes. Proc. 13th Annual Symposium on Fault-Tolerant Computing. 222–226 Milan. Italy, June 1983.
8. ARCHAMBEAU, E. C.–McCLUSKEY, E. J.: Fault Coverage of Pseudo-Exhaustive Testing. Proc. 14th Annual Int. Symp. Fault Tolerant Computing. 141–145. Orlando, FL. June 20–22, 1981.
10. WANG, L. T.: A New Condensed Linear Feedback Shift Register Design for VLSI/System Testing. Center for Reliable Computing Technical Report, Computer System Lab., Stanford University. 1984.
11. HASSAN, S. Z.–McCLUSKEY, E. J.: Pseudo-Exhaustive Testing of Sequential Machines Using Signature Analysis. Proc. 14th Annual Int. Symp. Fault Tolerant Computing, 360–365. Orlando, FL. June 20–22, 1984.
12. BRAZILAI, Z.–COPPERSMITH, D.–ROSENBERG, A.: Exhaustive Generation of Bit Patterns with applications to VLSI Self-Testing. IEEE Trans. Comput. vol. C-*32*, 190–194 (1983).
13. BARDELL, P. H.–McANNEY, W. H.: A View from the Trenches: Production Testing of a Family of VLSI Multichip Modules. Proc. 11th Annual Int. Symp. Fault-Tolerant Computing. Portland, ME. 281–283, June 24–26, 1981.
14. ICHIKAWA, M.: Constant Weight Code Generators. Center for Reliable Computing Technical Report, Computer System Lab., Stanford University. June 1982.
15. AGARWAL, V. K.–FUNG, A. S.: Multiple Fault Testing of Large Scale Circuits by Single Fault Test Sets. IEEE Trans. Comput. vol. C-*30*, 855–865 (1981).
16. CHANDRA, A. K.–KOU, L. T.–MARKOWSKY, G.: On Sets of Boolean n-Vectors with A11 k-Projections Surjective. IBM Res. Rep. RC-8936. July 1981.

Dr. Endre SELÉNYI H-1521 Budapest